Practical Experience Report (PER)

Attack detection without attack knowledge: reality or myth? Approaches and their application.

Abstract— **Anomaly-based intrusion detectors are machine learners trained to distinguish between normal and anomalous data. The normal data is generally easy to collect when building the train set; instead, collecting anomalous data requires historical data or penetration testing campaigns. Unfortunately, the first is most often unavailable or unusable, and the latter is usually expensive and unfeasible, as it requires hacking the target system. It turns out that the possibility of training an intrusion detector without attack knowledge, i.e., without anomalies, is attractive. This paper reviews strategies to train anomaly detectors in the absence of anomalies, from shallow machine learning to deep learning and computer vision approaches, and applies such strategies to the domain of intrusion detection. We experimentally show that training an intrusion detector without attack knowledge is effective when normal and attack data distributions are distinguishable. Detection performance severely drops in the case of complex (but more realistic) datasets, making all the existing solutions inadequate for real applications. However, the recent advancements of out-of-distribution research in deep learning and computer vision show interesting prospective results.**

Keywords—anomaly detection, intrusion detection, unknown attacks, out-of-distribution, autoencoder, GAN.

I. INTRODUCTION

Intrusion Detection Systems (IDSs) are becoming a fundamental building block of networked systems [1], [64], [65]. IDSs collect data from network and system indicators, and analyze them to detect malicious or unauthorized activities.

Signature-based IDSs rely on a dataset of signatures of known attacks, and continuously scan the network and host activities for signature matching. For example, to detect a malformed HTTP header attack, the headers of incoming packets are checked against reference values. Signature-based detectors perform satisfactorily when aiming at known attacks [64], [65]. However, they are weak against zero-day attacks [66]: these attacks exploit either new vulnerabilities or known vulnerabilities in novel ways and cannot be matched to known signatures. This problem cannot be neglected: zero-day attacks against a target system are a credible possibility, as confirmed by multiple reports in recent years [66], [76].

An attractive alternative to signature-based IDSs is anomaly-based IDSs, which observe the symptoms of an attack. In anomaly-based intrusion detection, the problem of detecting attacks is reduced to the problem of finding anomalies, i.e., patterns that do not conform to the expected behavior [56]. The underlying assumption is that attacks have a visible effect on the target system: it is possible to detect an ongoing attack by observing the value change of network and system indicators. Compared to signature-based IDSs, anomaly-based IDSs are more capable of detecting zero-day attacks, albeit their detection performance is still far from perfect [59]. Usually, anomaly-based IDSs rely on Machine Learning (ML) algorithms, where the anomaly detectors learn a model using a train set composed of both normal data and anomalous data.

A relevant criticality to the wide application of anomaly-based IDSs is the availability of such anomalous data, which should be retrieved either from historical data logs, or through the execution of penetration testing campaigns. Historical data is most often not available, or insufficient to perform a proper training of the ML algorithm. Penetration testing campaigns are often unfeasible, because of at least three reasons. First, penetration testing is costly and time-consuming. Second, it requires hacking the target system, which may be unacceptable. Last, it is impossible to assume that all possible attacks have been exercised: attacks may be left out from penetration tests and remain unknown to the intrusion detector.

There is evidence that the ability to train an IDS in the absence of attack knowledge is desirable, for example, when penetration testing is not performed, and attack data logs are not available. While solutions for training in the absence of anomalies have been investigated in the literature, their potential application to the intrusion detection domain is unclear, and a comparative evaluation is missing.

In this experience report, we explore and compare approaches to train an IDS in the absence of attack knowledge. First, we analyze unsupervised and semi-supervised algorithms, which can operate in the boundary condition of the absence of anomalous data in the train set. Then, we explore alternative solutions that rely on the generation of attack data. In particular, we explore out-of-distribution (OOD, [2]) generation and detection techniques that have been devised for tabular data and computer vision. Broadly, the goal is that the distribution of the generated data overrides the distribution of the real anomalous data, such that the intrusion detector can be trained using the generated data instead of the real anomalous ones. The key steps are i) generate data that does not belong to the normal data distribution, ii) label the generated data as anomalous and add it to the training set. Next, the ML algorithm is trained to distinguish the two distributions (the one from the normal data, and the one from the generated out-of-distribution data), and it will be naturally able to distinguish between normal and anomalous data.

More precisely, we identify multiple approaches, that comprise: i) the application of one-class anomaly detection algorithms, that trains using only normal data; ii) the

generation of OOD data to enrich the train set; iii) the application of OOD detectors based on GANs and autoencoders; iv) the application of data generation techniques from computer vision, which is a domain where data generation has been largely explored. We experiment with two public intrusion detection datasets, which are widely used in the literature and allow comparisons with respect to existing studies. Results show that IDSs without any attack knowledge show satisfactory detection performance when the distributions of normal and attack data are clearly distinguishable. In the case of complex (but more realistic) data flows, their detection performance severely drops, to the extent that they are unsuited for attack detection. However, the application of recent deep learning solutions based on data generation largely outperforms shallow (i.e., non-deep) machine learning approaches. This leads to two interesting remarks. First, it is one of the rare cases in which deep learning approaches significantly outperform shallow machine learning on tabular data [44], and, to our knowledge, the first case in the intrusion detection domain [59]. Second, as techniques for OOD in computer vision are constantly improving [62], [63], [68], their application to the purpose of intrusion detection is a promising research direction that can improve the results obtainable with the current technology.

The rest of the paper is organized as follows. In Section II we review OOD generation and detection, and we identify the relevant works that will be exercised in the rest of the paper. In Section III we define five strategies to train intrusion detectors in the absence of attack knowledge, and in Section IV we describe the implementation and application to two datasets. Results are in Section V and concluding remarks are in Section VI.

II. Background and Relevant Works on OOD

Usually, anomaly detectors are organized in supervised, unsupervised, and semi-supervised binary classifiers [56]. Supervised classifiers are trained using a labeled training set, where both normal and anomalies are available; unsupervised classifiers do not use labels during training, but anomalies are still present in the train set; finally, semi-supervised classifiers require that only a subset of the normal data is labeled.

When no anomalous data is available for training, some methods attempt to generate boundary data in the low-density region, or similarly, high-confidence Out-of-Distribution (OOD) data [5].

OOD detection can be formulated as a canonical binary classification problem. Let us consider an in-distribution P_X defined on an input space $X \in \mathbb{R}^n$. An OOD detector G: $X \rightarrow \{0, 1\}$ is built to distinguish whether an input $x \in X$ belongs to P_X or not. In the former case, the classifier assigns label 0 to x; conversely, it assigns label 1. During testing, the classifier G is evaluated on inputs drawn from a mixture distribution $M_{X \times Z}$ defined on $X \times \{0, 1\}$, where the conditional probability distributions $M_{X|Z=0} = P_X$ and $M_{X|Z=1} = Q_X$ [2]. Also, a common assumption is that P_X and Q_X are sufficiently different (can be distinguished) [2], [3], [4].

OOD generation and detection are typically applied to unstructured data, and the vast majority of applications are on images in the computer vision domain. Typically, their ultimate goal is to improve the robustness of a Deep Neural Network (DNN) model [73] or to protect against adversarial attacks (images purposedly altered to fail a trained model [6]).

OOD solutions and applications to tabular data [5] are a small number, and more specifically, the application to intrusion detection datasets is reduced to just a few cases.

In the following of this Section, we review related works on tabular data, which we will later exploit to develop OOD strategies for intrusion detection. We organize the related works in:

1. one-class anomaly detection algorithms, which are a special category of algorithms that train using only normal data. They aim to model the latent space of normal data (a latent space is the representation of items in a space of reduced dimensionality, where items resembling each other are positioned closer to one another [80]) without knowledge of the attacks in the dataset. Related works on this subject are in Section II.A.
2. generators of synthetic data, as similar as possible to the available data. Related works on this subject are in Section II.B.
3. generators of the effects of unknown attacks, to train OOD detectors using normal and synthetic attack data. Related works on this subject are in Section II.C.

A. One-class anomaly detection

While we are aware that the terminology in the literature is often non-univocal, we refer to the seminal works of Chen et al. [25], and Tax et al. [26], and we term *one-class anomaly detection* when classifiers are trained only with observations describing the normal behavior: one-class anomaly detection uses training data from only a single known class. Then, anomalies are data points differing from the distribution of the normal data: an observation is an anomaly if its distance from the normal data is above a threshold. This is independent of the density region that is created by the new observations.

One-Class Support Vector Machines (SVM, [25]) are the most notorious example. These can be defined as variants of linear and logistic regression models in which the notion of margin is used to avoid overfitting, just as regularization is used in regression models [33]. Several works have advanced SVM, for example, the Deep Support Vector Data Description (DeepSVDD, [32]) algorithm trains a neural network while minimizing the volume of a hypersphere that encloses the network representations of the data, forcing the network to extract the common factors of variation.

Recent advances in one-class anomaly detection are based on autoencoder architectures. An autoencoder is a neural network that first learns low-dimensional representations of the input data and then tries to reconstruct the high-dimensional representation of the input from such a reduced encoding. The lower the reconstruction error, the more likely the input data belongs to a known distribution [27]. In other words, an autoencoder learns the distribution of the normal data: novel inputs corresponding to the normal behavior are expected to be reconstructed accurately, while anomalous data tends to produce high reconstruction errors. Thus, the reconstruction error of the autoencoder can be used to perform anomaly detection.

Amongst the most famous approaches, the Outlier Detection AutoEncoder (OD-AE) is a reference autoencoder for anomaly detection presented in [33]. Instead, the Variational Autoencoder (VAE) from [34] optimizes a recognition model to perform approximate posterior inference using ancestral sampling.

Noticeably, many unsupervised classifiers (that require no labels at train time) can be trained using normal data only, albeit this is not the typical intended usage for which unsupervised learning is intended; possible examples are Isolation Forests (IF, [45]) and Clustering Based Local Outlier Factor (CBLOF) [46], which are notorious algorithms that usually have the highest performances amongst unsupervised approaches on tabular data [47].

Specifically for the case of intrusion detection, some examples of one-class anomaly detection using autoencoders are present in the literature. An autoencoder-based IDS is capable of detecting unknown types of attacks when their patterns deviate from the learned normal patterns [28]: the authors of [30], [31], and [36] train an IDS using the normal network traffic only, and test on the NSL-KDD or KDD Cup 99 datasets [42]. However, implementations to execute the algorithms are not available, and the datasets are hardly relevant as they are roughly 15 and 25 years old.

Other works operate from a different perspective. For example, the authors of [35] consider a semi-supervised approach, where training data has labeled points only for the normal class, using a dataset collected from multiple cyber-physical systems. As this work assumes that attacks are known and anomalies are present in the train set, it is considered out of scope for our research.

Considering the notoriousness of the algorithms in the tabular data domain, and the availability of implementations, *we exercise in this work SVM, IF, CBLOF, DeepSVDD, OD-AE, and VAE.*

B. Generation of In-Distribution Tabular Data

The problem of generating additional in-distribution tabular data can be described as follows: given a tabular dataset as input, we aim at generating data points that belong to the distribution of normal data, without being an exact duplicate of existing data points.

1) Generative Adversarial Networks (GANs)

The generation of synthetic data typically leverages Generative Adversarial Networks (GANs, [39]), which are capable of generating tabular data similar to the input data [10]. GANs simultaneously train two models: a generator and a discriminator. The generator aims at capturing the data distribution, while the discriminator aims at estimating the probability that an example comes from the training data rather than from the generated data. The discriminator outputs the probability that the input sample belongs to the same distribution of the real data rather than belonging to the distribution generated by the generator [39].

The vast majority of the research on GANs is in the computer vision domain [39]. GANs for tabular data are still in their early days [13], with few available libraries or meaningful contributions [7], [8], [13]. One of the first works to propose a GAN for tabular data is Xu et al. [7], where the authors present Tabular GAN (TGAN). TGAN enables the generation of tabular data with discrete and continuous variable types through an encoder-decoder LSTM (Long Short-Term Memory) recurrent neural network, which is a type of neural network designed to address sequence-to-sequence problems (given an input sequence, for example, a sentence, it produces an output sequence, where the number of items in the input and output sequences can vary). Instead, the authors of [8] present the Conditional Tabular GAN (CTGAN), a GAN-based method to model tabular data distribution and sample in-distribution data points. In CTGAN, the authors: i) propose a normalization step to overcome the non-Gaussian and multimodal distribution of tabular data, ii) design a conditional generator and train-by-sampling to deal with the discrete columns, and iii) train the model using a variational autoencoder modified for tabular data. Very recently, the authors of [13] present GANBLR (Generative Adversarial Network modeling inspired by Naive Bayes and Logistic Regression's relationship), which emphasizes the interpretability of the data generation models.

Considering the availability of libraries and documentation*, we will apply CTGAN and TGAN.*

2) Statistical Analysis

Very few approaches rely on statistical analysis to generate in-distribution tabular data. In our review of the recent literature, we identify only the approach in [9], termed Gaussian Copulas. A copula is a mathematical function that describes the joint distribution of multiple random variables by analyzing the dependencies between their marginal distributions (marginal distributions are the probability distribution of the variables contained in the subsets).

The key intuition underlying the copula function is that marginal distributions can be modeled independently from the joint distribution. Given a dataset with N features, a copula-based modeling approach would first model the N features independently, transforming them into uniform distributions, and then model the relationship between the transformed variables using the copula function. The approach proposed in [9] supports several multivariate distributions, and it can sample from them using copula functions and generate new synthetic data following the same statistical properties.

We will apply Gaussian Copulas in this work.

C. Generation of OOD Tabular Data

In the broad domain of ML, several works craft artificial OOD samples starting from in-distribution normal data. While the vast majority of applications are for computer vision, OOD generation can be used for a wide variety of tasks, and in the case of intrusion detection, it can simulate the effects of unknown or zero-day attacks. In fact, the generated OOD data can enhance the training process of anomaly-based intrusion detectors.

1) Generative Adversarial Networks (GANs)

The overwhelming majority of research on OOD GANs is focused on computer vision, and their application to tabular data is only limited to exemplifying cases [17], [18], [19], [20]. In general, these works take advantage of GANs to generate images on the tail of the data distribution. Often, the approach includes an autoencoder, which takes the place of the generator to reconstruct the original image. Then, the discriminator compares the original image with its

reconstruction: the reconstruction error is used to determine if the image is anomalous [21].

Some works that exploit GAN exist in the intrusion detection domain, but they use known anomalies in the train set to generate data similar to such anomalies [22], [23], [24]. In other words, they aim to increment the knowledge of known attacks. This way, the dataset is enriched with newly generated data that, in some cases, also enable the detection of zero-day attacks [23]. As these works require the availability of attack data in the train set, they are outside the scope of this paper.

Instead, out of the few related works on OOD tabular data generation, ARN (Adversarial Regularized Reconstruction, [15]) and ALAD (Adversarially Learned Anomaly Detection [16]) are recent proposals. Unlike other notorious techniques such as AnoGAN [40], or GANomaly [41], ARN and ALAD have been applied to tabular data and, most notably, tested on intrusion detection datasets. This makes them ready to operate in the domain considered in this paper. For this reason, we will favor ARN and ALAD over AnoGAN and GANomaly.

More precisely, ARN combines variational autoencoders and adversarial learning to generate realistic outliers for training the anomaly detector. The ARN autoencoder maps each sample in a compressed representation, from which an alternative reconstruction can be obtained with small differences from the original sample. The mapping with such compressed representation is controlled by exploiting regularization schemes (regularization in machine learning allows improving the generalization performance of a model). The reconstruction process enables the generation of data that, in the latent space, is relating to the abnormal behavior (or that it is sufficiently distant from the normal data).

Instead, ALAD exercises a bi-directional GAN, where the generator not only maps latent samples to generated data, but also has an inverse mapping from data to the latent representation [57]. As in ARN, computing the approximate latent representation for a data point x is done by passing x through an encoder E. ALAD also incorporates recent research results to improve the encoder by adding a discriminator to reduce cycle consistency loss [58], i.e., to improve the ability to reconstruct the input x after the $E(x)$ transformation.

We will use ARN and ALAD in the rest of this work.

2) Geometrical Methods

Only very few works heavily use geometrical methods to generate OOD data. We identified Soft Brownian Offset (SBO, [14]), which is an iterative approach to translate a point $x \in X$ by a distance d from a given dataset X, such that the translated point is an OOD data point. SBO is based on Gaussian Hyperspheric Offset, a common baseline to create OOD samples by sampling along a hypersphere normally distributed around the in-distribution data. Briefly, first, a suitable representation of an in-distribution dataset is created using an autoencoder. Then, data is translated using the SBO. After translation, the decoder of the autoencoder allows for the generation of these implicit out-of-distribution OOD samples. This way, new OOD data (anomalies) is synthetically generated. The OOD generation is regulated by two parameters, i) a minimum distance and ii) a softness value that determine respectively the minimum translation from the original point, and the ability to creep in the in-distribution values.

Specifically to the domain of intrusion detection, to the best of our knowledge, there are no intrusion detection works that generate OOD data using geometrical methods.

We will use Soft Brownian Offset in the rest of this work.

III. Detect Intrusions without Attack Knowledge

A. Five Strategies for Intrusion Detection

Building upon the analysis in Section II, we devise five strategies S1 to S5 to train intrusion detectors in the absence of attack knowledge, i.e., without anomalies in the train set.

S1. **One-class anomaly detection**. This is the straightforward application of one-class anomaly detection algorithms reviewed in Section II.A. These algorithms can be applied to intrusion detection datasets as well as to any other tabular dataset.

S2. **Imperfect generation of in-distribution tabular data**. Imperfect data, generated using techniques for in-distribution data generation, can augment the train set. This strategy is explained in Section III.B.

S3. **OOD generators for tabular data**. Techniques for generating OOD data generation can provide additional training data. This strategy is discussed in Section III.C.

S4. **Detectors that exploit OOD generation to support the training phase**. This approach is the direct application of existing frameworks for OOD detection. The reconstruction-based anomaly detectors ALAD and ARN previously reviewed match this strategy.

S5. **Rudimental application of computer vision techniques**. We transform tabular data points into images and feed them to image-based GANs to generate and detect OOD images. This strategy is discussed in Section III.D.

An additional possibility is the application of transfer learning, where a model is first trained on a dataset that includes anomalies, and then re-trained on a target dataset. This is a concrete possibility, for example, if a public intrusion detection dataset exists with the same features as the target dataset. However, the authors of [67] already demonstrated that this approach is not effective: they conclude that an IDS model trained to detect a given attack is not able to reveal variants of the same attack. In addition, a minor change with respect to the data collection environment of the train set invalidates patterns learned by the model. Consequently, a transfer learning strategy is not introduced in this paper.

B. S2: Imperfect generation of in-distribution tabular data

Generation of in-distribution data may fail when the target distribution is too difficult to learn or when the availability of in-distribution data is scarce.

This leads to the generation of data points that do not belong to the target distribution, even if it was the intended goal. When using GANs, this can be verified by observing the loss of the discriminator and the generator, which indicates the ability to generate data points close to the real data points. In a more general case, measuring the distance

between the real in-distribution data points and the generated data points provides a quantitative way to understand if generated data is in-distribution or OOD. In the latter case, we label the generated data as attack, and create a new train set that is comprised of both the real normal data and the generated OOD (attack) data.

Very practically, we compute the t-SNE distance (t-distributed stochastic neighbor embedding, [60]) between generated and normal data points. T-SNE is a nonlinear dimensionality reduction technique to represent high-dimensional data in a latent space of two or three dimensions, where data points resembling each other are positioned closer to one another. Given the t-SNE reduction, the distance between two data points can be computed as Euclidean distance. If the distance between a generated data point and the normal data point exceeds a given threshold, we consider the generated data point as anomalous and we use it to augment the train set. Otherwise, the generated data point is discarded. This way, we create a labeled train set that can be used to train supervised and unsupervised binary classifiers.

C. S3: OOD generators for tabular data

The reviewed SBO, ARN, and ALAD methods generate OOD tabular data. With SBO, the generation process depends solely on the configuration parameters, whereas for ARN and ALAD it also depends on the training epochs.

Similarly to S2, the application of OOD generators allows creating a new train set that is composed of normal data and of the generated anomalous data. This train set can be used to train supervised and unsupervised binary classifiers.

D. S5: Application of basic computer vision techniques

Strategies S1 to S4 rely on solutions from the tabular data domain. Since most of the research on OOD is in the domain of computer vision, we propose a strategy S5 that applies basic techniques of image OOD generation and detection to tabular data.

First, we convert the available normal data points into black-and-white images. This means that each data point is reshaped in a bi-dimensional matrix whose values are normalized between 0 and 1. The rationale is that tabular data transformed to bi-dimensional matrixes can be analyzed using DNN models for image classification, with good results [43]. In fact, the convolutional layers of a DNN model capture the relations between features and seize the important information. Our transformation is simply: i) apply a normalization step, ii) reshape the image, and iii) add 0-padding to match the desired image dimension.

After the tabular data is transformed into images, we apply a GAN for the generation of OOD images. At this point, the process is similar to S3 (see Section III.C). The generator and discriminator are trained for multiple epochs, saving small samples of generated data (i.e., images) at each checkpoint. When the generator learns how to successfully fool the discriminator, or when a target number of training epochs is reached, the training stops.

The generated images that have been saved are labeled as attacks and added to the train set, which can be used to train supervised and unsupervised algorithms.

Alternatively, we notice that at the end of the training process the discriminator learned to distinguish between normal and anomalous images, and consequently can act as an intrusion detector. Obviously, in case of a real deployment, the monitored tabular data should be converted to images and then fed to the discriminator. This can be easily implemented by adding two layers to the discriminator, that perform normalization and reshaping.

TABLE I. DETAILS OF THE INTRUSION DETECTION DATASETS. IN STRATEGIES S1-S5, THE TEST SET IS COMPOSED BY ALL THE ANOMALOUS DATA POINTS AND 40% OF THE NORMAL DATA POINTS.

Dataset Name	Year	Types of attacks	Features (columns)	Number of data points	Normal data points	Attack (anomalous) data points
ADFANet	2012	5	3	132 000	91 039	40 961
CICIDS18	2018	8	79	200 000	67 494	132 503

IV. EXPERIMENTS PLANNING AND EXECUTION

We set up an experimental campaign to quantify and compare the detection performance of the five strategies S1 to S5. The detection performance of mechanisms belonging to each strategy will be benchmarked against baseline intrusion detectors that use normal data and real anomalous data at train time.

A. Public Attack Datasets and Pre-Processing

The datasets we use are summarized in Table I. ADFANet (2012) [74] consists of both normal and anomalous (attacks) obtained from the TCP dump collected by Cisco routers. The occurrence of five different attacks is labeled. In ADFANet, the features are numeric and the analysis does not significantly benefit from one-hot encoding. However, both options with and without one-hot encoding are tested, and the best alternative is selected. One-hot encoding is applied to the feature named "packets", which has the lowest cardinality.

In Figure 1, we visualize the latent space of 30 000 normal data and 30 000 anomalous data from the test set in a 2D scatter plot. The reduction to two dimensions is realized by computing the t-SNE distance of the 60 000 data points. In this figure, perplexity, which is related to the number of nearest neighbors that each point has, is set to 20. Although different perplexity values lead to different visual effects, t-SNE is declared fairly robust to perplexity variation, and typical perplexity values are between 5 and 50 [60]. The clear separation of normal and anomalous data is evident, with most of the normal data grouped in a big cluster. As it will be later confirmed by our results, this gives us evidence that it is easy to discern the normal and anomalous data in ADFANet.

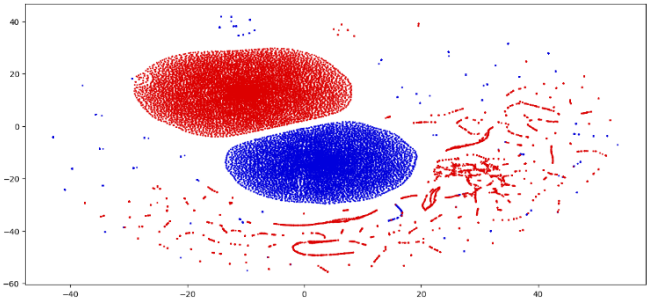


Figure 1. Bi-dimensional visualization of 60 000 elements from ADFANet. Normal data points are in blue, and anomalous data points are in red. Perplexity is set to 20. *Best viewed in color.*

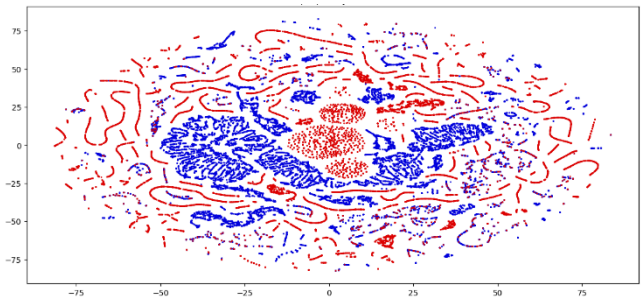


Figure 2. Bi-dimensional visualization of 50 000 elements from CICIDS18. Normal data points are in blue, and anomalous data points are in red. Perplexity is set to 20. *Best viewed in color.*

The dataset CICIDS18 [38] is collected by monitoring a real network topology which includes modems, firewalls, switches, routers, and nodes with different operating systems (Windows 10, Windows 8, Windows 7, Windows XP, iOS, and Linux). The dataset has 79 features and 8 different attacks. When pre-processing, we set the occurrences of Not-a-Number (nan) and infinite values to -1. We remove the timestamp and IP features because they implicitly differentiate sequences of anomalous data from normal data and consequently act as a hidden label once the dataset is shuffled. This is a common practice when operating with intrusion detection datasets [59].

Further, we one-hot encode categorical features. The criteria we apply to consider a feature as categorical are the feature semantics and the feature cardinality in the train set. We select 20 categorical features: Protocol, Fwd_PSH_Flags, Bwd_PSH_Flags, Fwd_URG_Flags, Bwd_URG_Flags, FIN_Flag_Cnt, SYN_Flag_Cnt, RST_Flag_Cnt, PSH_Flag_Cnt, ACK_Flag_Cnt, URG_Flag_Cnt, CWE_Flag_Count, ECE_Flag_Cnt, DownUp_Ratio, Fwd_Bytsb_Avg, Fwd_Pktsb_Avg, Fwd_Blk_Rate_Avg, Bwd_Bytsb_Avg, Bwd_Pktsb_Avg, Bwd_Blk_Rate_Avg. In total, the one-hot encoding adds 40 new features to the CICIDS18 dataset.

In Figure 2, we represent the t-SNE of 25 000 normal data and 25 000 anomalous data from the CICIDS18 test set. With respect to Figure 1, it is clear that the boundaries between the normal and anomalous data are not well-distinguishable, making the detection of attacks on CICIDS18 more difficult than on ADFANet.

Both datasets use a 60-40 train-test split. When needed to execute the strategies S1 to S5, anomalies are removed from the train set and added to the test set. When the strategy requires to generate anomalies, we generate 50 000 data points and add them to the train set.

B. Metrics for Unbalanced Datasets

The cardinality of normal and anomalous data in the test sets of ADFANet and CICIDS18 is significantly different. This is evident from the above consideration on the composition of the test sets and from Table I. In case of unbalanced datasets, a recommended metric is the Matthew Coefficient Correlation [54], [55]. It ranges between -1 and 1, where 0 means random guessing, 1 indicates that the prediction matches the ground truth, and -1 indicates that the prediction are opposite to the ground truth.

Given the confusion matrix true positives TP, false positives FP, true negatives TN, false negatives FN, the Matthew Coefficient Correlation MCC is:

$$\mathrm{MCC} = \frac{(\mathrm{TP}\cdot\mathrm{TN}) - (\mathrm{FP}\cdot\mathrm{FN})}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}$$

We also compute the accuracy, because it is widely used in anomaly detection and accounts for all the items of the confusion matrix. Accuracy ACC is defined as:

$$\mathrm{ACC} = \frac{\mathrm{TP}+\mathrm{TN}}{TP+FP+FN+TN}$$

C. Baseline anomaly detectors

We select well-known supervised and unsupervised anomaly detectors as baseline: they provide reference values, i.e., state-of-the-art results on the considered datasets. In the following, we term *baseline* the supervised and unsupervised classifiers trained on both normal and anomalous data.

The selected supervised baselines are XGBoost (XGB, [48]) and Random Forests (RF, [49]), which are respectively boosting and bagging ensembles of decision trees. They are very effective for intrusion detection [75], [79], and they usually outperform neural networks when processing tabular data [44]. As unsupervised algorithms, we use Isolation Forests (IF) [45] and CBLOF [46], which are amongst the most known unsupervised classifiers and find wide application in anomaly detection tasks. Implementation of RF and IF is from scikit-learn [72], while implementation of XGB and CBLOF is respectively from [71] and [47].

Clearly, we expect that baseline anomaly detectors achieve the best detection performance, because they rely on knowledge of both normal data and attacks at train time.

D. Implementation of Strategies S1 to S5

We detail the implementation of strategies S1 to S5 with the aid of Table II and Figure 3. Table II reports the main techniques that implement each strategy and the baseline. Strategies S2, S3, and S5 are divided into a generation step and a training step: the table reports both the data generation technique and the ML algorithm applied afterward. Further, Table II includes acronyms introduced to facilitate readability through the rest of the paper, and information on the train set. Figure 3 visually reports the process that implements the baseline and the strategies.

Implementation of S1. The implementation of SVM, OD-AE, VAE, CBLOF, DeepSVDD is from [47], while the implementation of IF is from [72]. S1 straightforwardly applies the above libraries and does not require further discussion.

Implementation of S2. We use TabGAN, CTGAN, and GC as generators of in-distribution data. Their implementation is respectively from [78], [50], and [51]. For each technique, we try multiple configurations and training epochs to find the best possible tuning. More precisely:

- TabGAN is trained for 10 epochs on ADFANet, while on CICIDS18 it is trained for 20 epochs with an upper bound of 0.2 to the Gaussian noise added to the categorical columns.
- CTGAN trains for 20 epochs with default parameters.

- With GC, we use the Gaussian multivariate setting, which operates multiple random variables at the same time and takes into account the dependencies that may exist between them.

For each technique, the generated data is logged. Then, data is processed using t-SNE to remove data points that are too close to the normal data and cannot be considered OOD. This requires defining a proximity threshold: we test with multiple thresholds and, in this paper, we report the results of the ones leading to the highest MCC.

This process allows obtaining enriched train sets composed of normal data plus generated data. Then, baseline classifiers can be trained on such train sets: we will show results for XGB, IF, and CBLOF.

Implementation of S3. We use three different OOD generators, namely SBO, ARN, and ALAD. Their implementation is respectively from [52], [53], and [77]. The setup is as follows.

SBO is trained with 2 500 combinations of minimum distances and softness values (see SBO description in Section II). This way, data points are generated and collected from each configuration, to maximize the distribution of the data points in the latent space. The minimum distances are 50 values uniformly distributed in the interval [0.6, 1.0] for ADFANet, and in the interval [0.3, 1.0] for CICIDS18. The softness values are 50 uniformly distributed in the interval [0.4, 1.0] for ADFANet, and [0.2, 1.0] for CICIDS18.

ARN is trained for 400 epochs. The implementation of ARN provided by the authors requires one-hot encoding of at least one feature, and data normalization.

ALAD is trained for 35 epochs; the weight of the sum of the mapping loss function is set to 0.1.

Generated data is logged into separate files for each of the three techniques. These are used to create three train sets composed of normal data plus data generated with each of the three techniques. Next, binary classifiers from the baseline can be exercised: we will show results for XGB, IF, and CBLOF.

Implementation of S4. ARN and ALAD are configured as in S3. Their execution to act as OOD detectors does not

TABLE II. REVIEW OF THE BASELINE CLASSIFIERS AND THE FIVE STRATEGIES. WE REPORT THE MAIN TECHNIQUE THAT IMPLEMENTS EACH STRATEGY, IF THERE IS A CLASSIFIER TRAINED ON THE GENERATED ANOMALIES, AND IF THE REAL ANOMALIES ARE IN THE TRAIN SET.

Strategy		Technique (acronym)	Algorithm trained on generated anomalies	Original anomalies in train set
Baseline	supervised	RandomForest (RF) XGBoost (XGB)	--	YES
	unsupervised	CBLOF, IsolationForest (IF), AutoEncoder (OD-AE), SVM, VAE	--	YES
S1	one-class anomaly detection	CBLOF, DeepSVDD, IF, OD-AE, SVM, VAE	--	NO
S2	Imperfect generation of in-distribution tabular data	CTGAN	XGB CBLOF IF	NO
		TabGAN	XGB CBLOF IF	
		Gaussian Copulas (GC)	XGB CBLOF IF	
S3	OOD generators for tabular data	ALAD	XGB CBLOF IF	NO
		ARN	XGB CBLOF IF	
		SBO	XGB CBLOF IF	
S4	Detectors that include OOD generation	ALAD ARN	--	NO
S5	Computer vision OOD	GAN for images	Decision Tree	NO

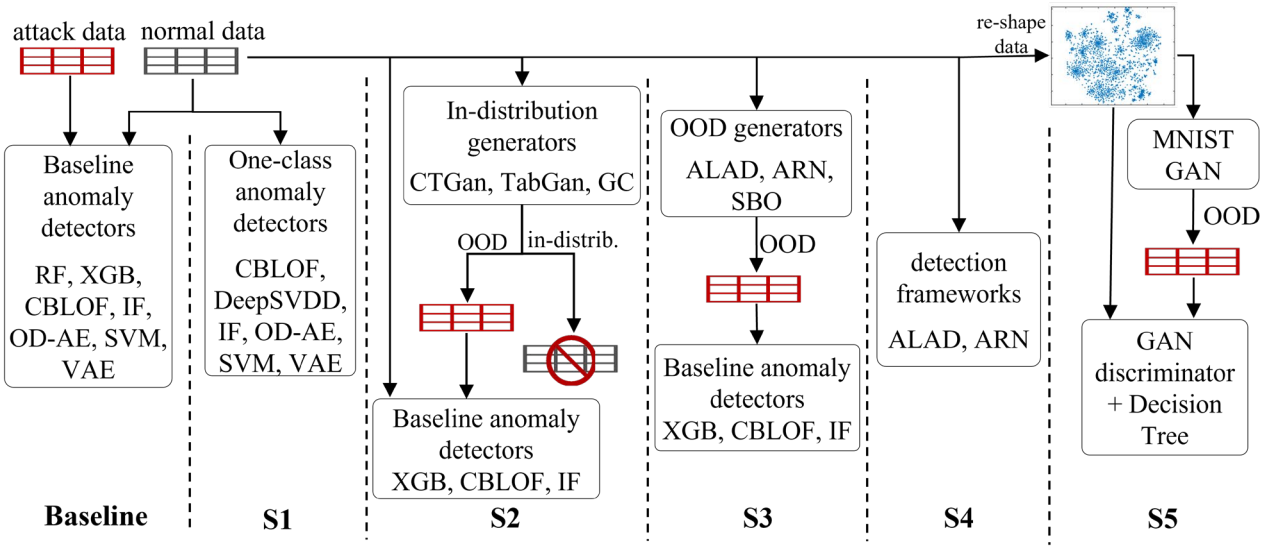


Figure 3. Application of baseline classifiers and strategies S1 to S5. The acronyms are explained in Table II.

require additional setup parameters, and will not be discussed further.

Implementation of S5. We use the tutorial GAN from TensorFlow available at [29]. This GAN operates on the MNIST dataset, which is composed of black-and-white images of 28x28 shape. We reshape tabular data points into a 28x28 array, with zero padding where needed. The GAN employs a generator with 3 convolutional layers, 1 dense layer and above 2 000 000 trainable parameters, and a discriminator with 2 convolutional layers, 1 dense layer, and above 200 000 trainable parameters.

After training, we try two different approaches to detect anomalies, and we select the one leading to the highest MCC: i) proceed as in S2 to generate data and train an anomaly detector; ii) use the trained discriminator as an anomaly detector. In this second case, the discriminator of our GAN outputs a positive number instead of a binary label and thus it cannot be directly used to detect intrusions. We overcome this problem by connecting the discriminator output to a decision tree: the decision tree will learn the threshold to transform the discriminator score into a binary label, enabling binary classification. This second approach offered the best results.

E. Execution, data availability, and reproducibility

The code to reproduce the experiments is available at [37]. It consists of two Jupyter notebooks respectively for ADFANet and CICIDS18, plus two modified versions of ALAD and ARN software, adapted to operate with the new datasets. To promote reproducibility, the train and test split that we used are also available at [37].

Experiments have been executed on a Dell Precision 5820 Tower with an Intel Xeon Gold 6250, GPU NVIDIA Quadro RTX6000 with 24GB VRAM, 192GB RAM and Ubuntu 18.04, NVIDIA driver 45.119.03 with CUDA 11.

V. Results

Results are summarized in Table III and discussed below.

A. ADFANet

Not surprisingly, the highest MCC is achieved with the baseline supervised classifiers: both XGB and RF have excellent MCC and accuracy. Baseline unsupervised classifiers have variable results, with MCC ranging from 0.337 (SVM) to 0.872 (CBLOF), and are significantly worse than baseline supervised algorithms. This is expected, and has been repeatedly observed in intrusion detection works [12], [11].

One-class anomaly detection (S1) approaches reach an interesting MCC=0.959 with CBLOF. This is a higher MCC than with baseline unsupervised algorithms. For example, the baseline CBLOF (i.e., CBLOF executed with attacks known at training time) has MCC=0.872, which is lower than the S1 CBLOF (MCC=0.959), trained without knowledge of the attacks. This peculiar result can be explained first remembering that ADFANet is a simple dataset where it is easy to discern anomalous data from normal data. In S1, CBLOF can precisely identify the clusters and their boundaries, and consequently, it can separate normal from anomalous data. Since the normal data points are well grouped, creating this boundary is easy and effective. Instead, in the baseline CBLOF, the data contamination (the presence of anomalous data in the train set) confuses the algorithm, which is not able to distinguish which clusters are of normal data and which are of anomalous data. In fact, this is a more difficult task than the previous one (given the reduced complexity of ADFANet). Similar reasoning applies to IF, SVM, DeepSVDD, OD-AE, and VAE algorithms.

The other baseline or S1 approaches have worse performances than CBLOF, although they all show good capabilities of discerning between attacks and normal data, with few exceptions. SVM is significantly worse than the others, but it is now superseded by more performing solutions, as also evident from our results.

S2 solutions reach results similar to the baseline CBLOF, especially with TabGAN (MCC=0.95). It should be remarked that many data points generated by TabGAN and CTGAN are identical to normal data points in the train set, and consequently are discarded. ADFANet is composed of three features, which are integer values: thus, it is easy to understand which generated data is too close to the normal

TABLE III. ACCURACY ACC AND MATTHEW CORRELATION COEFFICIENT MCC ON ADFANET AND CICIDS18. IN BOLD ARE THE BEST RESULTS OF THE BASELINE AND EACH STRATEGY.

	Strategy	Technique	Algorithm trained on generated anomalies	ADFANet		CICIDS18	
				ACC	MCC	ACC	MCC
Baseline	sup.	RF		**0.999**	**0.997**	0.961	0.914
		XGB		**0.999**	**0.997**	**0.966**	**0.928**
	Unsupervised	CBLOF		0.946	0.872	0.858	0.676
		IF		0.784	0.453	0.549	0.084
		OD-AE		0.895	0.747	0.656	0.272
		SVM		0.655	0.337	0.575	0.127
		VAE		0.886	0.728	0.658	0.275
S1	one-class anomaly detection	CBLOF		**0.979**	**0.959**	0.686	0.209
		DeepSVDD		0.654	0.390	**0.752**	**0.479**
		IF		0.973	0.948	0.629	0.012
		SVM		0.687	0.369	0.575	0.127
		OD-AE		0.903	0.809	0.414	0.086
		VAE		0.939	0.878	0.415	0.092
S2	Imperfect generation of in-distribution tabular data	CTGAN	XGB	0.5	0.165	0.169	0.000
			CBLOF	0.919	0.846	0.835	0.177
			IF	0.957	0.916	0.357	0.129
		TabGAN	XGB	0.599	0.218	0.169	0.000
			CBLOF	**0.975**	**0.950**	0.336	0.034
			IF	0.918	0.846	0.462	0.146
		GC	XGB	0.597	0.357	0.169	0.000
			CBLOF	0.668	0.444	0.728	0.035
			IF	0.964	0.929	**0.501**	**0.199**
S3	OOD generators for tabular data	ALAD	XGB	0.471	0.000	0.169	0.000
			CBLOF	0.668	0.444	**0.849**	**0.313**
			IF	0.973	0.946	0.497	0.183
		ARN	XGB	0.471	0.027	0.290	0.000
			CBLOF	0.667	0.422	0.416	0.071
			IF	0.667	0.422	0.572	0.269
		SBO	XGB	0.612	0.381	0.169	0.000
			CBLOF	0.922	0.851	0.441	0.158
			IF	**0.975**	**0.950**	0.834	0.132
S4	Include OOD generation	ALAD		0.643	0.372	0.392	0.226
		ARN		**0.995**	**0.990**	**0.780**	**0.622**
S5	Computer vision OOD	GAN	Decision Tree	**0.668**	**0.464**	**0.865**	**0.580**

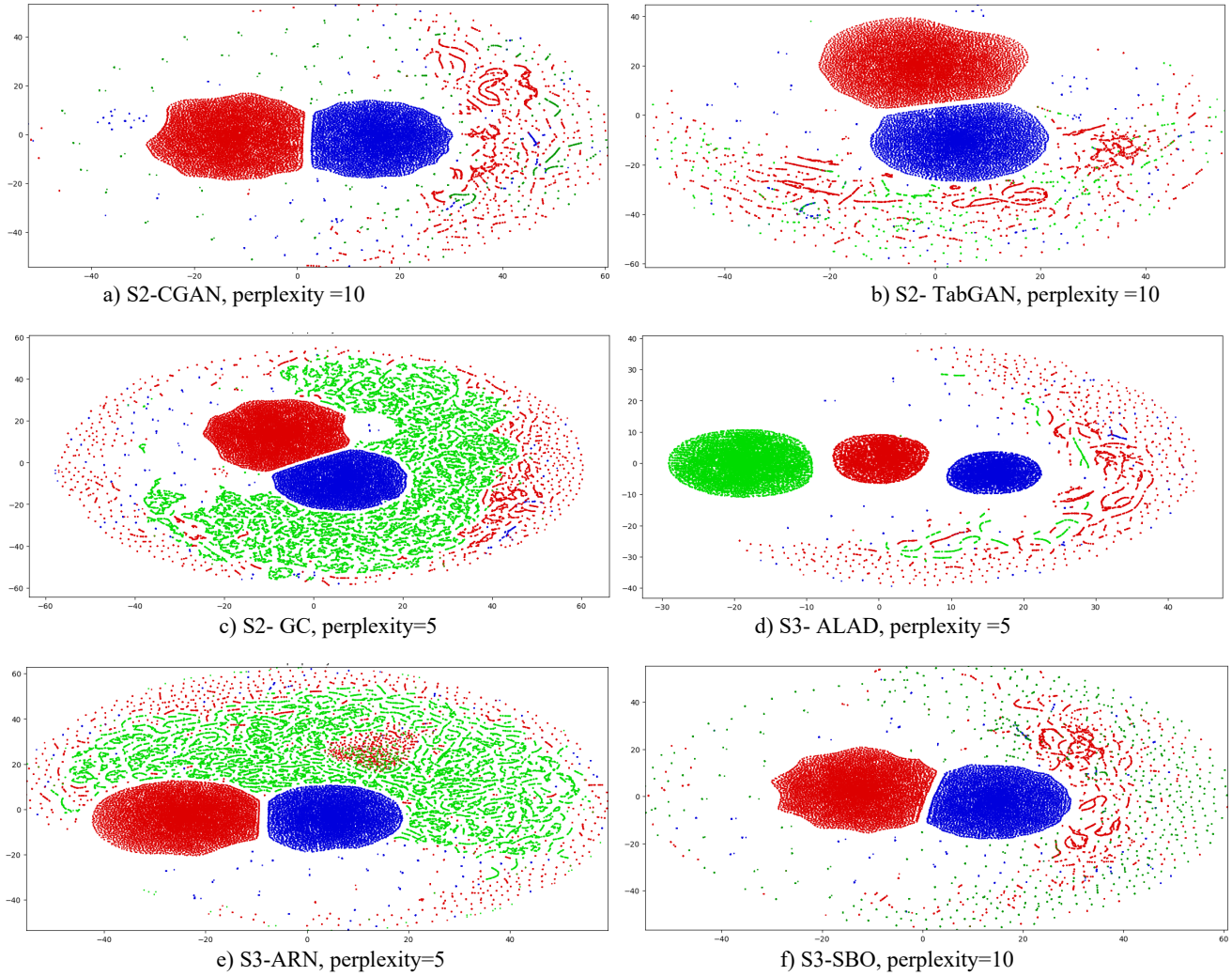
data, and discard it. This instead does not apply to GC, which generates data farther from the normal distribution than TabGAN and CTGAN.

We represent and further explain the data generation of TabGAN, CTGAN, and GC with the aid of Figure 4. The figure shows the t-SNE plot of ADFANet, where normal data is in blue, real anomalous data is in red, and generated anomalous data is in green. Normal and anomalous data are the same as in Figure 2, but they are depicted differently because the generated data takes part to the dimensionality reduction process. Looking at Figure 4a and Figure 4b, we observe how only a few sparse (green) data points are generated by TGAN and CTGAN. Instead, Figure 4c (GC) has much more green data points (generated anomalies), but they are concentrated in a well-defined portion of the latent space, and the overlap with the red data points (real anomalous data) is very small.

S3 solutions reach results similar to the baseline CBLOF, especially with IF applied on a train set enriched with data generated with SBO (MCC=0.95) and ALAD (MCC=0.94). Instead, XGB applied on the same train set is not effective at all, with MCC close to 0 (MCC=0 is random guessing). Most likely, this is because XGB tries to learn a very precise characterization of the normal and the generated anomalous data: since the overlap between the clusters of generated anomalous data and real anomalous data is minimal, the learning process of XGB does not converge with an obvious performance drop. Conversely, unsupervised algorithms usually create looser characterization than supervised algorithms and have better detection performance in this case. Figure 4c, Figure 4d, and Figure 4e confirm our conjecture on data distribution: the distance between the generated data points and the others is evident, especially in the case of ALAD.

In S4, ARN shows a very good MCC (0.990), which is very close to the MCC of baseline supervised algorithms (0.997). This is an important result, because it shows that data generation is a feasible approach to mitigate the absence of anomalous data. This may be surprising, because the data generated by ARN (at the end of its training epochs) does not lead to good performance in S3, which may look contradictory: we believe this is due to the characteristics of the ARN generator which rapidly converge to little data variability, while the discriminator is progressively trained, and it cannot be considered a weakness of the generation algorithm.

Figure 4. t-SNE applied to ADFANet. Normal data points are in blue, anomalous data points are in red, and generated anomalous data points are in green. Different perplexity values are used, depending on which visualization is more explicative of the output. *(best viewed in colors)*



a) S2-CGAN, perplexity =10

b) S2- TabGAN, perplexity =10

c) S2- GC, perplexity=5

d) S3- ALAD, perplexity =5

e) S3-ARN, perplexity=5

f) S3-SBO, perplexity=10

Last, our strategy S5 performs poorly (MCC=0.464). This can be expected, because ADFANet has only three features of natural numbers, while we are using approaches for unstructured data that are thought for much higher entropy. In this case, strategies S1-S4 for tabular data are preferable.

Summarizing, although no generative solution sufficiently matches the latent space of anomalous data, as visualized in Figure 4, some strategies can reach satisfactory results, also to the extent that baseline supervised algorithms are almost matched.

B. CICIDS18

The supervised baseline has the highest scores, with MCC above 0.91 for both XGB and RF. The difference between the supervised and unsupervised baseline is significant, with CBLOF being the best-performing one (MCC=0.676). This confirms that the separability of normal and anomalous data in CICIDS18 is less evident than in ADFANet: the loose decision boundaries learned by unsupervised algorithms (even when all the attacks are included in the train set) are not adequate to detect intrusions in CICIDS18. All the other baseline algorithms have MCC below 0.3, which is considerably low.

This trend continues also with S1, S2, and S3 strategies, which offer MCC scores below 0.5. In particular, anomaly generators cannot create synthetic anomalous data that is close to the real anomalous data.

ARN in S4 is again the best approach, with MCC=0.622. This score is very far from the supervised approaches, but higher than all the unsupervised baselines. S4 can be considered a very promising strategy to train an intrusion detector in the absence of attack knowledge, although more improvements are needed to reach results of practical use.

The S5 strategy, with MCC=0.580, is promising as it outperforms all the other strategies except S4 ARN. This result can be justified by the fact that CICIDS18 has several features, and the transformation of tabular data points to images is in this case effective. Importantly, it should be considered that S5 uses very simple approaches from computer vision: considering the constant advancements of OOD in computer vision, there is probably room to apply novel OOD solutions and further improve the S5 results.

VI. Concluding Remarks

A. Threats to validity

Internal validity is concerned with factors that may have influenced the results, but they have not been thoroughly considered in the study. We have used two very different public datasets, that are extensively used in intrusion detection works. We selected algorithms implementations provided by the authors or available in notorious libraries that require minimal manipulation and as such reduce the risk of mistakes from our side. For each algorithm, we have performed tunings of parameters using grid searches, and selected those that led to the highest metric scores. Experiments were repeated multiple times, to guarantee that results are consistent through multiple runs.

External validity refers to what extent the results of the study can be applied to other domains. We are not aiming at claiming the validity of this study for domains other than intrusion detection. It is evident that the discussed solutions could be exercised to anomaly detection in other domains, but we cannot foresee the quality of the generated data or the performance of the ML algorithms.

Reproducibility. The usage of public data and public tools to run algorithms was a prerequisite of our analysis to allow reproducibility and to rely on existing implementations. In addition, we publicly share the code and data on github [37], allowing any researcher or practitioner to repeat the experiments.

B. Barriers to the practical application

Trust in the intrusion detector. It is difficult to trust the detection capability of a solution without having seen it in action. Very practically, before deploying an intrusion detector, a stakeholder may want guarantees of the ability to detect attacks. This means that, at least to a small extent, some realistic anomalous data must be available to test the intrusion detector.

Limited generalization of results. The research results of our experimental study depend on the datasets in use. Therefore, it is difficult to anticipate the possible results for a given dataset, without analyzing it as we did for ADFANet and CICIDS18. However, we are confident that the effectiveness of the identified strategies is strictly connected to the complexity of the datasets, and consequently, the main implications of this paper are valid in the case of datasets that have a similar representation of the latent space.

C. Conclusions and Future Works

This paper investigated solutions to build anomaly-based intrusion detectors in the absence of attack data at training time. Therefore, we i) reviewed techniques for anomaly and out-of-distribution (OOD) detection, ii) contextualized them to the domain of intrusion detection, and iii) crafted and exercised five different strategies.

The most effective strategies are based on deep learning solutions. This is an interesting outcome, because it has been repeatedly proved that shallow machine learning is better than deep learning on tabular data [44]. In addition, good results are achieved with basic techniques from computer vision, which is a promising alternative because computer vision already offers plenty of solutions to deal with OOD [62], and is constantly proposing new advancements that we hypothesize can be applied also to tabular data.

Consequently, our future works will further explore OOD approaches from computer vision. For example, the recent works [61], [63], [69], [70] present OOD techniques that have been crafted and applied to image and textual (language) datasets, but not to tabular datasets or intrusion detection datasets. We believe the application of such techniques in this domain can further improve the results we report in this paper.

Acknowledgment

Removed for double-blind.

References

[1] Haider, W., Hu, J., Slay, J., Turnbull, B. P., and Xie, Y. "Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling". Journal of Network and Computer Applications, 87, 185-192, 2017.

[2] Chen, J., Li, Y., Wu, X., Liang, Y., and Jha, S. "Robust out-of-distribution detection for neural networks". arXiv preprint arXiv:2003.09711, 2020.

[3] Bendale, A., and Boult, T. E. "Towards open set deep networks". In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1563-1572), 2016.

[4] Sehwag, V., et al. "Analyzing the robustness of open-world machine learning". Proceedings of the 1^{2}th ACM Workshop on Artificial Intelligence and Security. 2019.

[5] Yang, J., et al. "Generalized out-of-distribution detection: A survey". arXiv preprint arXiv:2110.11334. 2021.

[6] Biggio, B., et al. "Evasion attacks against machine learning at test time". In ECML PKDD 2013, Prague, Czech Republic, September 23-27, Proceedings, Part III 13 (pp. 387-402). Springer Berlin Heidelberg. 2013.

[7] Xu, L., and Veeramachaneni, K. "Synthesizing tabular data using generative adversarial networks". arXiv preprint arXiv:1811.11264. 2018.

[8] Xu, L., Skoularidou, M., Cuesta-Infante, A., and Veeramachaneni, K. "Modeling tabular data using conditional GAN". Advances in Neural Information Processing Systems, 32. 2019.

[9] Patki, N., Wedge, R., and Veeramachaneni, K. "The synthetic data vault". In 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA) (pp. 399-410). IEEE. 2016.

[10] Bourou, S., et al. "A review of tabular data synthesis using GANs on an IDS dataset". Information, 12(09), 375. 2021.

[11] Laskov, P., Düssel, P., Schäfer, C., and Rieck, K. "Learning intrusion detection: supervised or unsupervised?". In Image Analysis and Processing–ICIAP 2005, pp. 50-57. Springer Berlin Heidelberg. 2005.

[12] Zoppi, T., Ceccarelli, A., Puccetti, T., and Bondavalli, A. "Which Algorithm can Detect Unknown Attacks? Comparison of Supervised, Unsupervised and Meta-Learning Algorithms for Intrusion Detection". Computers & Security, 103107. 2023.

[13] Zhang, Yishou, et al. "Interpretable tabular data generation". Knowledge and Information Systems (2023): 1-29.

[14] Moller, F., Botache, D., Huseljic, D., Heidecker, F., Bieshaar, M., and Sick, B. "Out-of-distribution detection and generation using soft brownian offset sampling and autoencoders". In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 46-55). 2021.

[15] Liguori, A., Manco, G., Pisani, F. S., and Ritacco, E. "Adversarial Regularized Reconstruction for Anomaly Detection and Generation". In IEEE International Conference on Data Mining (ICDM), pp. 1204-1209. IEEE. 2021.

[16] Zenati, H., et al. "Adversarially learned anomaly detection". IEEE International Conference on Data Mining (ICDM). IEEE, 2018.

[17] Wolleb, J., Sandkühler, R., and Cattin, P. C. Descargan. "Disease-specific anomaly detection with weak supervision". In Medical Image Computing and Computer Assisted Intervention–MICCAI 2020, pp. 14-24. Springer International Publishing. 2020.

[18] Du, Xuefeng, et al. "Siren: Shaping representations for detecting out-of-distribution objects". Advances in Neural Information Processing Systems. 2022.

[19] Marek, P., Naik, V. I., Auvray, V., and Goyal, A. "Oodgan: Generative adversarial network for out-of-domain data generation". arXiv preprint arXiv:2104.02484. 2021.

[20] Dionelis, N., Yaghoobi, M., and Tsaftaris, S. A. "Tail of distribution GAN (TailGAN): Generative Adversarial-network-based boundary formation". In 2020 Sensor Signal Processing for Defence Conference (SSPD), pp. 1-5. IEEE. 2020.

[21] Ran, X., et al. "Detecting out-of-distribution samples via variational auto-encoder with reliable uncertainty estimation". Neural Networks, 145, 199-208. 2022.

[22] Shahriar, M. H., Haque, N. I., Rahman, M. A., and Alonso, M. "G-ids: Generative adversarial networks assisted intrusion detection system". In IEEE ^{4}4th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 376-385. IEEE. 2020.

[23] Kim, J. Y., Bu, S. J., and Cho, S. B. "Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders". Information Sciences, 460, 83-102. 2018.

[24] Lin, Z., Shi, Y., and Xue, Z. "Idsgan: Generative adversarial networks for attack generation against intrusion detection". In Advances in Knowledge Discovery and Data Mining, May 16–19, 2022, Springer International Publishing. 2022.

[25] Chen, Y., Zhou, X. S., and Huang, T. S. "One-class SVM for learning in image retrieval". In International Conference on Image Processing, Vol. 1, pp. 34-37. IEEE. 2001.

[26] Tax, D., and Duin, R. "Support vector data description". Machine learning 54 (2004): 45-66.

[27] Lo, S.-Y., Oza, P., and Patel, V. "Adversarially Robust One-class Novelty Detection". IEEE Transactions on Pattern Analysis and Machine Intelligence. 2022.

[28] Song, Y., Hyun, S., and Cheong, Y. G. "Analysis of autoencoders for network intrusion detection". Sensors, 21(13), 4294. 2021.

[29] Tensorflow tutorial on GANs, https://www.tensorflow.org/tutorials/generative/dcgan

[30] Gharib, M., Mohammadi, B., Dastgerdi, S. H., and Sabokrou, M. "Autoids: Auto-encoder based method for intrusion detection system". arXiv preprint arXiv:1911.03306. 2019.

[31] Aygun, R. C., and Yavuz, A. G. "Network anomaly detection with stochastically improved autoencoder based models". In 2017 IEE^{E} ^{4t}h international conference on cyber security and cloud computing (CSCloud) (pp. 193-198). IEEE. 2017.

[32] Lukas Ruff, et al. "Deep one-class classification". International conference on machine learning. 2018.

[33] Aggarwal, C. "Outlier analysis". In *Data mining*, 75–79. Springer, 2015.

[34] Diederik P. K., and Welling, M. "Auto-encoding variational bayes". arXiv preprint arXiv:1312.6114. 2013.

[35] Catillo, M., Pecchia, A., and Villano, U. "CPS-GUARD: Intrusion detection for cyber-physical systems and IoT devices using outlier-aware deep autoencoders". Computers & Security 129 (2023): 103210.

[36] Farahnakian, F., and Heikkonen, J. "A deep auto-encoder based approach for intrusion detection system". International Conference on Advanced Communication Technology (ICACT). IEEE. 2018.

[37] Anonymous github (online): https://anonymous.4open.science/r/attack_generation-210A

[38] Sharafaldin, I., Lashkari, A., and Ghorbani, A. "Toward generating a new intrusion detection dataset and intrusion traffic characterization". ICISSP, pp. 108-116. 2018.

[39] Xia, X. et al. "GAN-based anomaly detection: a review". Neurocomputing. 2022.

[40] Schlegl, T., et al. "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery". International Conference on Information Processing in Medical Imaging, pp. 146–157. 2017.

[41] Akcay, S., Atapour-Abarghouei, A., and Breckon, T. "Ganomaly: Semi-supervised anomaly detection via adversarial training". In ACCV 2018, Revised Selected Papers, Part III, pp. 622-637. Springer International Publishing. 2019.

[42] Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. "A detailed analysis of the KDD CUP 99 data set". In 2009 IEEE symposium on computational intelligence for security and defense applications, pp. 1-6. IEEE. 2009.

[43] Sharma, Alok, et al. "DeepInsight: a methodology to transform a non-image data to an image for convolution neural network architecture". Scientific reports 9.1 (2019): 11399.

[44] Shwartz-Ziv, R., and Armon, A. "Tabular data: deep learning is not all you need". Information Fusion, 81, 84-90. 2022.

[45] Liu, F. T., Ting, K. M., and Zhou, Z. H. "Isolation forest". In IEEE International Conference on Data Mining, pp. 413-422. IEEE. 2008.

[46] Gao, Z. "Application of cluster-based local outlier factor algorithm in anti-money laundering". In International Conference on Management and Service Science, pp. 1-4. IEEE. 2009.

[47] Zhao, Y., Nasrullah, Z. and Li, Z. "PyOD: A Python Toolbox for Scalable Outlier Detection". Journal of machine learning research (JMLR), 20(96), pp.1-7. 2019.

[48] Chen, T., and Guestrin, C. "Xgboost: A scalable tree boosting system." Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016.

[49] Breiman, L. "Random forests". Machine learning, 45, 5-32. 2001.

[50] CTGAN implementation, https://github.com/sdv-dev/CTGAN

[51] Gaussian copulas implementation, https://github.com/sdv-dev/Copulas

[52] Soft Brownian Offset (SBO) implementation, https://github.com/flxai/soft-brownian-offset

[53] Adversarial Regularized Reconstruction (ARN) implementation, https://github.com/arnwg/arn

[54] Chicco, D., and Jurman, G. "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". BMC genomics, 21, 1-13. 2020.

[55] Boughorbel, S., Jarray, F., and El-Anbari, M. "Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric". PloS one, 12(6), e0177678. 2017.

[56] Chandola, V., Banerjee, A., and Kumar, V.. "Anomaly detection: a survey". ACM computing surveys (CSUR) 41.3 (2009): 1-58.

[57] Donahue, J., Krähenbühl, P., and Darrell, T. "Adversarial feature learning". arXiv preprint arXiv:1605.09782. 2016.

[58] Zhu, J.-Y., et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". Proceedings of the IEEE international conference on computer vision. 2017.

[59] Zoppi, T., Ceccarelli, A., and Bondavalli, A. "Unsupervised algorithms to detect zero-day attacks: strategy and application". IEEE Access 9 (2021): 90603-90615.

[60] Van der Maaten, L., and Hinton, G. "Visualizing data using t-SNE". Journal of machine learning research 9.11. 2008.

[61] Amit, G., et al. "FOOD: Fast out-of-distribution detector". International Joint Conference on Neural Networks (IJCNN). IEEE. 2021.

[62] Fort, S., Ren, J., and Lakshminarayanan, B. "Exploring the limits of out-of-distribution detection". Advances in Neural Information Processing Systems 34 (2021): 7068-7081.

[63] Kirchheim, K., Filax, M., and Ortmeier, F. "Pytorch-ood: A library for out-of-distribution detection based on pytorch". Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022.

[64] Zhao, Z., et al. "Robust anomaly detection on unreliable data". IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 2019.

[65] Palmieri, F. "Network anomaly detection based on logistic regression of nonlinear chaotic invariants". Journal of Network and Computer Applications 148 (2019): 102460.

[66] Bilge, L., and Dumitraş, T. "Before we knew it: an empirical study of zero-day attacks in the real world". Proceedings of the 2012 ACM conference on Computer and communications security. 2012.

[67] Catillo, M., Del Vecchio, A., Pecchia, A., and Villano, U. "Transferability of machine learning models learned from public intrusion detection datasets: the CICIDS2017 case study". Software Quality Journal, 1-27. 2022.

[68] Salehi, M., et al. "A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges". arXiv preprint arXiv:2110.14051. 2021.

[69] Du, X., et al. "Vos: Learning what you don't know by virtual outlier synthesis". arXiv preprint arXiv:2202.01197. 2022.

[70] Dionelis, N., Yaghoobi, M., and Tsaftaris, S. "Omasgan: Out-of-distribution minimum anomaly score gan for sample generation on the boundary". arXiv preprint arXiv:2110.15273. 2021.

[71] XGBoost library, https://xgboost.readthedocs.io/.

[72] Sci-kit learn library, https://scikit-learn.org/stable/.

[73] Lin, R., et al. "Robustness evaluation for deep neural networks via mutation decision boundaries analysis" Information Sciences 601: 147-161. 2022.

[74] Tran, Q. A., Jiang, F., and Hu, J. "A real-time netflow-based intrusion detection system with improved BBNN and high-frequency field programmable gate arrays". In 11th International Conference on Trust, Security and Privacy in Computing and Communications. IEEE. 2012.

[75] Jiang, H., et al. "Network intrusion detection based on PSO-XGBoost model". IEEE Access, 8, 58392-58401. 2020.

[76] Meakins, J. "A zero-sum game: the zero-day market in 2018". Journal of Cyber Policy, 4(1), 60-71. 2019.

[77] Adversarially learned anomaly detection (ALAD) implementation, https://github.com/houssamzenati/Efficient-GAN-Anomaly-Detection

[78] TGAN implementation, https://github.com/sdv-dev/TGAN

[79] Resende, P., and Drummond, A. "A survey of random forest based methods for intrusion detection systems." ACM Computing Surveys (CSUR), 51(3), 1-36. 2018.

[80] Liu, Y., et al. "Latent space cartography: Visual analysis of vector space embeddings." Computer graphics forum. Vol. 38. No. 3. 2019.